PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-024929
(43)Date of publication of application : 29.01.1999

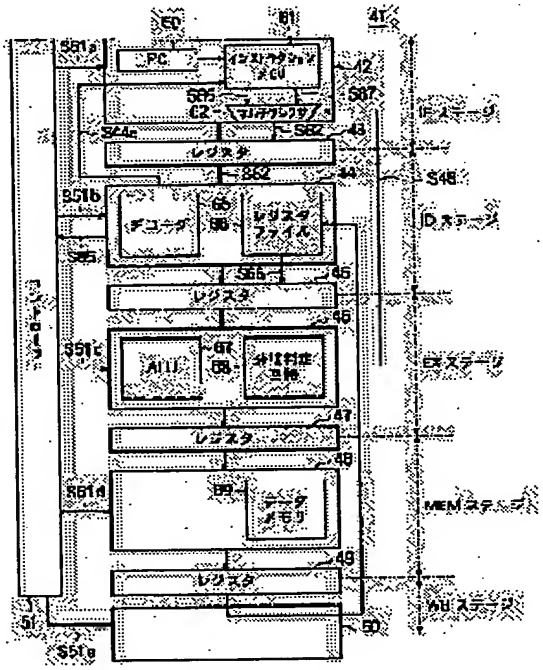
(51)Int.Cl. G06F 9/38

(21)Application number : 09-174407
(22)Date of filing : 30.06.1997

(71)Applicant : SONY CORP
(72)Inventor : HANAKI HIROICHI

(54) ARITHMETIC PROCESSING UNIT AND ITS METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To provide an arithmetic processing unit which can effectively suppress the deterioration of a processing efficiency owing to a branch instruction in a pipeline processing.
SOLUTION: When a branch instruction is recognized as a result of decoding an instruction at an ID module 44, the presence or absence of branching is judged at an EX module 46 in a subsequent cycle, and the instruction of a branch destination and the instruction of a non-branch destination are simultaneously fetched at an IF module 42. The fetched instruction of the branch destination or the instruction of the non-branch destination is selected based on the judged result of the presence or absence of branching. And the selected instruction is decoded at the ID module 44.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

CLAIMS

[Claim(s)]

[Claim 1] Fetch processing of the instruction memorized at least by the instruction storage section by the fetch means, Decoding of said fetched instruction by the decoding means, and data processing according to the result of said decoding by the operation means, In the processing unit which performs storage processing as a result of said data processing by the storage processing means by the pipelined architecture said fetch means The program counter which increments the address on said instruction storage section one by one, and points to it, The address storage section which memorizes the address of the branching place contained in the branch instruction concerned when the decoded instruction is branch instruction, The instruction storage section which memorizes an instruction to two or more bank fields accessible to coincidence, The instruction memorized to the address on said instruction storage section to which said program counter points when the decoded instruction was branch instruction, The fetch section which fetches to coincidence the instruction memorized to the address on said instruction storage section to which it points with the address memorized by said address storage section, The processing unit which has the selection section which chooses one instruction among the instructions fetched to said coincidence based on the judgment result of the branch condition about said decision instruction, and is outputted to said decoding means.

[Claim 2] The processing unit according to claim 1 memorized on the bank where the instruction of the branching place by the decision instruction differs from the instruction of an un-branching place in said instruction storage section.

[Claim 3] The processing unit according to claim 1 with which a number of said bank field of instructions continuously processed one by one by said instruction storage section are memorized to a different bank field.

[Claim 4] Said instruction storage section is a processing unit according to claim 1 which is single port memory with a singular read-out port.

[Claim 5] It is the processing unit according to claim 1 which has further

the flag storage section which memorizes the flag which shows that said fetch means has the effective address memorized by said address storage section, and fetches the instruction memorized to the address on said instruction-storage section to which it points with the address memorized by said address storage section only when it is shown that said fetch section has the effective flag memorized by said flag storage section.

[Claim 6] Said fetch section of said fetch means is a processing unit according to claim 1 which pinpoints said bank field by the 1st field of said address, and specifies the address in a bank field by the 2nd field of said address.

[Claim 7] Said decoding means is a processing unit according to claim 1 which has the decoding section which decodes the instruction chosen by said selection section and generates the control signal for an instruction execution, and the data storage section which memorizes the data used in an operation means.

[Claim 8] Said operation means is a processing unit according to claim 1 which has arithmetic Boolean part and the branching judging section which judges the branch condition of branch instruction.

[Claim 9] A storage processing means is a processing unit according to claim 1 which performs processing memorized in the data storage section which builds in the result of said data processing, and processing which memorizes the result of said data processing in the data storage section of said decoding means.

[Claim 10] The processing unit according to claim 1 which has said fetch means, a decoding means, a singular operation means, and a singular storage processing means.

[Claim 11] Fetch processing of the instruction memorized by the instruction storage section at least and said fetched decoding of an instruction, In the data-processing approach of performing data processing according to the result of said decoding, and storage processing as a result of said data processing by the pipelined architecture Increment the address on said instruction storage section one by one, and it points to the address of an instruction of an un-branching place. When the decoded instruction is branch instruction, the address of the branching place contained in the branch instruction concerned is memorized. To the instruction storage section which equipped coincidence with two or more accessible bank fields, the instruction of a branching place and the instruction of an un-branching place are memorized to a different bank field. The instruction of the un-branching place memorized to the address on said said instruction storage section to which it pointed when the decoded instruction

was branch instruction, The instruction of the branching place memorized to the address on said instruction storage section to which it points with the address memorized by said address storage section is fetched to coincidence. The data-processing approach which chooses and decodes one instruction among the instructions fetched to said coincidence based on the judgment result of the branch condition about said decision instruction.

DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]

[Field of the Invention] This invention relates to a processing unit and its approach.

[0002]

[Description of the Prior Art] Generally, as the RISC (Reduced Instruction Set Computer) processor built in DSP (Digital Signal Processor) etc. is shown below, it performs data processing according to a program. That is, at a processor, data processing of the instruction is performing about each of the instruction included in a program by performing the (WB) stage which writes a result in the (IF) stage which fetches an instruction from instruction memory, the (ID) stage which decodes the fetched instruction (decode), the (EX) stage which executes the decoded instruction, the stage (MEM) which accesses memory, and the memory which accessed in order, for example. In this case, when timing which fetches the next instruction is made into the timing which WB stage about a previous instruction ended, the time amount which doubled total of the time amount spent on each of IF stage, ID stage, EX stage, an MEM stage, and WB stage from the timing which starts the fetch of a previous instruction to the timing which WB stage of the next instruction ends two is needed. Drawing 6 is the block diagram of the processor 1 of the conventional computer. As shown in drawing 6 , a processor 1 has the IF module 2, a register 3, the ID module 4, a register 5, the EX module 6, a register 7, the MEM module 8, a register 9, the WB module 10, and a controller 11.

[0003] The IF module 2, the ID module 4, the EX module 6, the MEM module 8, and the WB module 10 perform IF stage, ID stage, EX stage, an MEM stage, and WB stage, respectively.

[0004] Here, from the former, by the processor 1, in order to increase the amount of operations per unit time amount, the pipeline processing which calculates in juxtaposition each stage mentioned above is adopted about two or more instructions. In pipeline processing, as shown in drawing 7 , processing of each stage is ended in 1 cycle, an

instruction is inputted into a processor in order for every cycle, and IF stage, ID stage, EX stage, MEM stage, and WB stage of a different instruction are performed to juxtaposition.

[0005] By the processor 1, as shown in drawing 7 , instruction n-n +4 is inputted into a processor 1 at intervals of 1 cycle, and, specifically, WB stage of Instruction n, the MEM stage of instruction n+1, EX stage of instruction n+2, ID stage of instruction n+3, and IF stage of instruction n+4 are performed to juxtaposition in a cycle 20. Thus, if five steps of pipeline processing is adopted, compared with the case where pipeline processing is not adopted, the amount of operations of 1 cycle unit can be raised 5 times.

[0006] In addition, although illustrated about the case where five steps of pipeline processing is adopted, in the processor 1 mentioned above, it is also possible to subdivide processing of an instruction further, to raise a clock frequency by simplifying processing of each stage, and to increase further the amount of operations per unit time amount.

[0007]

[Problem(s) to be Solved by the Invention] As mentioned above, as shown in drawing 7 , when EX stage is started about Instruction n, by the processor 1, ID stage of instruction n+1 and IF stage of instruction n+2 are started. By the way, it is judged only after whether whether Instruction n being a decision instruction's branching although recognized on ID stage, and branch condition are fulfilled when Instruction n is a decision instruction processes Instruction n on EX stage. Therefore, when it becomes clear that Instruction n is branch instruction, the fetch of the instruction n+1 of consecutiveness of Instruction n and n+2 has already been carried out.

[0008] If it continues passing instruction n+1 and n+2 to a pipeline as they are at this time, the instruction (instruction arranged immediately after branch instruction) of an un-branching place will be executed, and right actuation cannot be performed. therefore -- for example, as shown in drawing 8 , when it is judged with branching on EX stage, the instruction n+1 of the already fetched consecutiveness and n+2 are discarded, and the instruction m of a branching place, m+1, and .. are fetched from the following cycle in order. However, when the instruction already fetched in this way is canceled, there is a problem that processing effectiveness will fall. For example, when shown in drawing 8 , two-cycle delay is carried out according to generating of branching. In order to solve such a problem, the instruction surely executed regardless of the judgment result of the existence of branching among the instructions which follow

branch instruction may be arranged immediately after branch instruction, and the technique of delay branching which is delayed and performs the instruction with which the existence of activation is determined by the existence of branching may be adopted. Here, the instruction group performed regardless of the existence of branching among the instructions which follow branch instruction is called a delay slot.

[0009] If the number of the instructions of a delay slot is more than the number of the instructions which may be discarded after a fetch is carried out by generating of branching when such delay branching is adopted, a delay slot can be embedded immediately after branch instruction, but when that is not right, it is necessary to embed the nop (no operation) instruction which nothing performs immediately after branch instruction, and there is a problem that processing effectiveness will fall. For example, in the example shown in drawing 7 , if the number of instructions of a delay slot does not have two or more, it will be necessary to embed a nop instruction and processing effectiveness will fall.

[0010] Moreover, on ID stage, when branch instruction has been recognized, after it stops and whether it branches or not determines a pipeline, there is a method of fetching the instruction of a branching place or an un-branching place, and making a pipeline restart etc.

[0011] However, by any approach, a pipeline will stop and processing effectiveness will fall until it specifies the instruction which should be carried out a fetch, since the instruction which should be carried out a fetch next until it executes a decision instruction (judgment of branching) cannot be specified. Thus, in the processor 1 which adopted pipeline processing, it is important how this is reduced by decision-instruction **** and a branching penalty existing, when raising processing effectiveness.

[0012] In order to reduce this branching penalty, there is also the approach of predicting beforehand whether it branches or not, but when prediction separates, a big penalty arises. Moreover, there is also a problem that a processor will large-scale-ize, by preparing the circuit for predicting. moreover, although there is also a method of performing a branching judging all over ID stage, and branching it immediately, the data set as the object of a judgment are calculating with the instruction before a decision instruction in that case (EX stage) -- etc. -- it becomes a critical path and high-speed mounting is difficult for a case.

[0013] This invention is made in view of the conventional technique mentioned above, and aims at offering the processing unit which can control effectively decline in the processing effectiveness by branch instruction, and its approach in pipeline processing.

[0014]

[Means for Solving the Problem] Solve, and in order to attain the purpose which mentioned above the trouble of the conventional technique mentioned above, the processing unit of this invention Fetch processing of the instruction memorized at least by the instruction storage section by the fetch means, Decoding of said fetched instruction by the decoding means, and data processing according to the result of said decoding by the operation means, It is the processing unit which performs storage processing as a result of said data processing by the storage processing means by the pipelined architecture. Said fetch means The program counter which increments the address on said instruction storage section one by one, and points to it, The address storage section which memorizes the address of the branching place contained in the branch instruction concerned when the decoded instruction is branch instruction, The instruction storage section which memorizes an instruction to two or more bank fields accessible to coincidence, The instruction memorized to the address on said instruction storage section to which said program counter points when the decoded instruction was branch instruction, The fetch section which fetches to coincidence the instruction memorized to the address on said instruction storage section to which it points with the address memorized by said address storage section, It has the selection section which chooses one instruction among the instructions fetched to said coincidence based on the judgment result of the branch condition about said decision instruction, and is outputted to said decoding means. Here, in said instruction storage section, the instruction of the branching place by the decision instruction and the instruction of an un-branching place are memorized on a different bank.

[0015] Moreover, fetch processing of an instruction in which the data-processing approach of this invention was memorized by the instruction storage section at least, Decoding of said fetched instruction, and data processing according to the result of said decoding, Are the data-processing approach of performing storage processing as a result of said data processing by the pipelined architecture, and the address on said instruction storage section is incremented one by one. Point to the address of an instruction of an un-branching place, and when the decoded instruction is branch instruction In the instruction storage section which memorized the address of the branching place contained in the decision instruction concerned, and equipped coincidence with two or more accessible bank fields The instruction of the un-branching place which memorized the

instruction of a branching place, and the instruction of an un-branching place to a different bank field, and was memorized to the address on said said instruction storage section to which it pointed when the decoded instruction was branch instruction, The instruction of the branching place memorized to the address on said instruction storage section to which it points with the address memorized by said address storage section is fetched to coincidence. Based on the judgment result of the branch condition about said decision instruction, one instruction is chosen and decoded among the instructions fetched to said coincidence.

[0016]

[Embodiment of the Invention] Hereafter, the processor concerning the operation gestalt of this invention is explained. Drawing 1 is the block diagram of the processor 41 of this operation gestalt. As shown in drawing 1, a processor 41 has the IF module 42, a register 43, the ID module 44, a register 45, the EX module 46, a register 47, the MEM module 48, a register 49, the WB module 50, and a controller 51.

[0017] The IF module 42, the ID module 44, the EX module 46, the MEM module 48, and the WB module 50 perform IF stage, ID stage, EX stage, an MEM stage, and WB stage, respectively. Like the processor 1 mentioned above, although a processor 41 performs pipeline processing, processings of branch instruction differ in a processor 1. That is, in a processor 41, like a processor 1, as shown in drawing 6, processing of each stage is ended in 1 cycle, an instruction is inputted into a processor for every cycle in order, IF stage, ID stage, EX stage, and the MEM stage of five instructions are performed to juxtaposition, and pipeline processing is performed. Moreover, in a processor 41, as a result of differing in a processor 1 and decoding an instruction in the ID module 44, when it has been recognized as it being a decision instruction, it is the following cycle, and while judging the existence of branching in the EX module 46, in the IF module 42, the instruction of a branching place and the instruction of an un-branching place are fetched to coincidence. And in the following cycle, based on the judgment result of the existence of branching, either the fetched instruction of a branching place and the instruction of an un-branching place are chosen, and this selected instruction is decoded in the ID module 44.

[0018] Hereafter, the component of a processor 41 shown in drawing 1 is explained to a detail. First, the IF module 42 is explained. As shown in drawing 1, the IF module 42 has the multiplexer 62 as a program counter 60, the instruction memory 61, and the selection section. A program counter 60 points to the address on the instruction memory 61 of the instruction which reads to a degree based

on control signal S51a from a controller 51, and increments the address one by one for every cycle.

[0019] Drawing 2 is the block diagram of the instruction memory 61. As shown in drawing 2 , the instruction memory 61 is the memory 80 as the instruction storage section, a flag register 81, address registers 82 and 83, the access-control section 841-848 as the fetch section, and a multiplexer 851-858. And it has multiplexers 86 and 87. Memory 80 is bank 801-808. It is the single port memory which has eight banks, and these eight banks can be accessed now at coincidence. Thus, an equipment configuration can be made small-scale and cheap by using single port memory as memory 80. In addition, as for the number of banks of memory 80, it is desirable to make it the exponentiation of 2.

[0020] bank 801-808 **** -- it is shown in drawing 3 -- as -- under a program -- instructions 1, 2, 3, 4, 5, 6, 7, and 8 -- order -- respectively -- bank 801, 802, 803, 804, 805, 806, 807, and 808 memorizing -- next, instruction 9 and ... bank 801 from -- bank 808 It goes and memorizes in order. When a decision instruction occurs by doing in this way, the probability for the instruction of a branching place and the instruction of an un-branching place to be memorized on the same bank is set to one eighth. If the instruction of a branching place and the instruction of an un-branching place are memorized on the same bank, it will become impossible that is, to fetch these instructions to coincidence. Thus, when the instruction of a branching place and the instruction of an un-branching place have been memorized on the same bank, either a decision instruction or the instruction of a branching place is replaced with other instructions so that the semantics (semantics) of a program may not change. When it is not made, a nop (empty) instruction is inserted, and the location of an instruction is shifted so that the instruction of a branching place and the instruction of an un-branching place may not be memorized on the same bank. Consequently, it becomes possible to be able to memorize the instruction of a branching place, and the instruction of an un-branching place on a different bank, and to read these instructions to coincidence about a decision instruction.

[0021] Thus, when memory 80 is made 8 bank configurations, the low order triplet of the address memorized by address registers 82 and 83 shows a bank number, and the bit of a high order shows the address in each bank from it. Bank 801-808 The low order triplet of the address memorized by an address register 82 or 83 becomes active at the time of "000", "001", "010", "011", "100", "101", and "111", respectively.

[0022] An address register 82 memorizes the address on the memory 80 the

instruction of the un-branching place to which a program counter 60 points was remembered to be. An address register 83 memorizes the address on the memory 80 the instruction of a branching place was remembered to be inputted from the ID module 44. Thus, by the IF module 42, in order to access two banks of memory 80 at coincidence, it has two address registers. A flag register 81 memorizes the flag which shows whether the address of the branching place memorized by the address register 83 is effective. When memorizing the address of a branching place from the ID module 44 to an address register 83, the flag which shows "1" is memorized, and when other, the flag which shows "0" is memorized by the flag register 81.

[0023] Multiplexer 851-858 For example, either is chosen among the address of an instruction of the un-branching place memorized by the address register 82 based on control signal S51a from a controller 51, and the address of an instruction of the branching place memorized by the address register 83, and it is the access-control section 841-848, respectively. It outputs. the access-control section 841-848 Bank 801-808 where the low order triplet of the address corresponds based on the address from a multiplexer 851-858, respectively the high order bit of the remainder of the address when shown -- using -- bank 801-808 from -- an instruction is read. Moreover, the access-control section 841-848 Bank 801-808 by the address memorized by the address register 83 when the flag memorized by the flag register 81 was "1" Read-out actuation is not performed.

[0024] a multiplexer 86 -- the access-control section 841-848 from -- bank 801-808 specified by the low order triplet of the address memorized by the address register 82 among read-out results from -- a read-out result is chosen and this selected branching place instruction S86 is outputted to a multiplexer 62. a multiplexer 86 -- the access-control section 841-848 from -- bank 801-808 specified by the low order triplet of the address memorized by the address register 83 among read-out results from -- a read-out result is chosen and this selected un-branching place instruction S87 is outputted to a multiplexer 62.

[0025] By the IF module 42, the branching place instruction to which it points in the address of the un-branching place instruction to which it points in the address memorized by the address register 82, and the address memorized by the address register 82 is read to coincidence. At this time, it is under judgment whether a decision instruction occurs on EX stage and branches. Since the branching judging result S46 is returned to a multiplexer 62 from the EX module 46 before this judgment cycle

finishes, either the branching place instruction S86 read to coincidence or the un-branching place instruction S87 is chosen in a multiplexer 62 by that result, and processing of the IF module 42 is ended. After this selected instruction S62 is latched with the register 43 shown in drawing 1 , it is outputted to the ID module 44.

[0026] Next, the ID module 44 shown in drawing 1 is explained. The ID module 44 has a decoder 65 and a register file 66, as shown in drawing 1 . A decoder 65 decodes the instruction S62 inputted from the IF module 42 through the register 43 based on control signal S51b. While creating various kinds of control signals for an instruction execution and outputting this control signal S65 to a controller 51 After accessing a register file 66, reading the data used for the operation in the EX module 46 and latching this read data S66 with a register 45, it outputs to the latter EX module 46. Moreover, a decoder 65 makes the flag register 81 of the IF module 42 memorize the flag which shows "1" while it outputs address S44a of a branching place to the address register 83 of the IF module 42 shown in drawing 2 and it makes it memorize, when it is a decision instruction, as a result of decoding the instruction S62 from a register 43. Thereby, in the following cycle, in the IF module 42, a branching place instruction and an un-branching place instruction are read to coincidence at the same time the branching judging of this decision instruction is carried out in the EX module 46.

[0027] Next, the EX module 46 shown in drawing 1 is explained. The EX module 46 has ALU (Arithmetic and Logic Unit)67 which performs data processing, the branching judging circuit 68, and the address-generation circuit which is not illustrated. ALU67 performs data processing using data S66 based on control signal S51c according to the decoding result from a controller 51. An address-generation circuit generates the address on the data memory 69 which memorizes the data of the data-processing result of ALU67. In addition, after ALU67 reads the data memorized by data memory 69 to a register file 66, it accesses a register file 66 and uses the data for data processing. Moreover, ALU67 memorizes a data-processing result to data memory 69 through a register file 66.

[0028] The EX module 46 outputs the data-processing result of ALU67, and the address which the address-generation circuit generated to the MEM module 48 through a register 47. The branching judging circuit 68 outputs the branching judging result S46 which the instruction under operation is branch instruction in ALU67, and directs to branch when it judges with branching, as a result of evaluating branch condition to the IF

module 42. The IF module 42 chooses and outputs either the branching place instruction fetched to coincidence, or a non-decision instruction by the multiplexer 62 shown in drawing 2 based on the branching judging result S46.

[0029] Next, the MEM module 48 is explained. The MEM module 48 has data memory 69 and the control circuit which is not illustrated. In a write-in instruction, the MEM module 48 memorizes the data of a data-processing result inputted into the address on the control signal S data memory 69 inputted from the EX module 46 through the register 47 based on 51d from a controller 51 from the EX module 46 (it writes in).

[0030] In a read-out instruction, the MEM module 48 reads data from the address on the control signal S data memory 69 inputted from the EX module 46 through the register 47 based on 51d from a controller 51. Moreover, in the instruction which does not access data memory 69, the MEM module 48 outputs the data of an input **** data-processing result as it is through a register 49 at the WB module 50 through a register 47 from the EX module 46.

[0031] Furthermore, based on the control signal from a controller 51, the MEM module 48 chooses either the data read from data memory 69, or the data of the result of an operation from the EX module 46 by the multiplexer, and outputs it to the WB module 50 through a register 49.

[0032] Next, the WB module 50 is explained. The WB module 50 memorizes the data inputted from the MEM module 48 to the register file 66 of the ID module 44 through a register 49 based on control signal S51e.

[0033] Hereafter, actuation of a processor 41 is explained. Drawing 4 is drawing for explaining pipeline processing when branching by branch instruction occurs in a processor 41. First, in a cycle "1", the fetch of the instruction n is carried out by the IF module 42 shown in drawing 1 , and in the following cycle "2", while decoding of Instruction n is performed by the ID module 44, the fetch of instruction n+1 is performed by the IF module 42. The access-control section 841-848 which the flag which shows "0" is memorized by the IF module 42 by the flag register 81 shown in drawing 2 at this time, and is shown in drawing 2 Based on the address to which the program counter 60 memorized by the address register 82 points, an instruction is read from memory 80 and this read instruction is outputted to a register 43 through multiplexers 86 and 62. Moreover, in the ID module 44, the decoder 65 which it is recognized that Instruction n is a decision instruction and it shows to drawing 1 memorizes the address of an instruction of a branching place to an address register 83 while memorizing the flag which shows "1" to the flag register 81 of

the instruction memory 61 shown in drawing 2 .

[0034] Next, in the cycle "3" shown in drawing 4 , when it is judged whether the branch condition of Instruction n is fulfilled, for example, branch condition is fulfilled in the branching judging circuit 68 of the EX module 46, it outputs to the multiplexer 62 which shows the branching judging result S46 which shows that to drawing 1 and drawing 2 . It is based on the address memorized by it and coincidence in the instruction memory 61 shown in drawing 2 at the address register 82 and the address register 83, and is the access-control section 841-848. It sets and the branching place instruction m and the un-branching place instruction n+2 are read from memory 80. And the un-branching place instruction n+1 (S86) and the branching place instruction m (S87) are outputted to a multiplexer 62, and in a multiplexer 62, based on the branching judging result S46, the branching place instruction m is chosen and it is outputted to the MEM module 48 through a register 47 as instruction S62. Moreover, the instruction n+1 by which the fetch was carried out by the IF module 42 in the cycle "2" is canceled.

[0035] Next, in a cycle "4", the MEM stage of Instruction n, ID stage of Instruction m, and IF stage of instruction m+1 are performed by the MEM module 48, the ID module 44, and the IF module 42, respectively.

[0036] Next, in a cycle "5", WB stage of Instruction n, EX stage of Instruction m, ID stage of instruction m+1, and IF stage of instruction m+2 are performed in the WB module 50, the EX module 46, the ID module 44, and the IF module 42, respectively. Hereafter, similarly, unless a decision instruction exists, IF stage, ID stage, EX stage, an MEM stage, and WB stage are performed one by one about instruction m+3, m+4, and ..

[0037] Drawing 5 is drawing for explaining pipeline processing in case branching by branch instruction does not occur in a processor 41. In this case, pipeline processing in case branching by the branch instruction shown in drawing 4 mentioned above generates a cycle "1" and "2", and same processing are performed.

[0038] Next, in a cycle "3", when it is judged whether the branch condition of Instruction n is fulfilled, for example, branch condition is not fulfilled in the branching judging circuit 68 of the EX module 46, it outputs to the multiplexer 62 which shows the branching judging result S46 which shows that to drawing 1 and drawing 2 . It is based on the address memorized by it and coincidence in the instruction memory 61 shown in drawing 2 at the address register 82 and the address register 83, and is the access-control section 841-848. It sets and the branching place instruction m and

the un-branching place instruction n+2 are read from memory 80. And the un-branching place instruction n+2 (S86) and the branching place instruction m (S87) are outputted to a multiplexer 62, and in a multiplexer 62, based on the branching judging result S46, the un-branching place instruction n+2 is chosen, and it is outputted to the MEM module 48 through a register 47 as instruction S62. Moreover, in a cycle 2, the instruction n+1 by which the fetch was carried out in the IF module 42 is canceled.

[0039] Next, in a cycle "4", the MEM stage of Instruction n, ID stage of instruction n+1, and IF stage of instruction n+2 are performed by the MEM module 48, the ID module 44, and the IF module 42, respectively.

[0040] Next, in a cycle "5", WB stage of Instruction n, EX stage of instruction n+1, ID stage of instruction n+2, and IF stage of instruction n+3 are performed in the WB module 50, the EX module 46, the ID module 44, and the IF module 42, respectively. Hereafter, similarly, unless a decision instruction exists, IF stage, ID stage, EX stage, an MEM stage, and WB stage are performed one by one about instruction n+4, n+5, and ..

[0041] As explained above, when it has been recognized as an instruction being a decision instruction in the ID module 44 according to the processor 41, while this decision instruction is executed and carrying out the branching judging in the EX module 46 in the following cycle, as soon as it has read both the branching place instruction and the un-branching place instruction to coincidence and the branching judging result S46 is obtained, the corresponding instruction is chosen by the IF module 42. Therefore, if the branching judging results S46 are any of branching and not branching, a branching place instruction or an un-branching place instruction can be outputted to the ID module 44 at the following cycle. Therefore, compared with the conventional parallel processor 1 mentioned above, decline in the processing effectiveness by branching generating can be controlled effectively.

[0042] Specifically, according to the processor 41, only the cycle according to the count of a branch instruction appearance can shorten the processing time compared with the technique of not performing the conventional branch prediction. Moreover, according to the processor 41, only the cycle according to the count from which the branch prediction separated compared with the technique of performing the conventional branch prediction can shorten the processing time. Moreover, according to the processor 41, only the number of the nop instructions inserted in order not to fill a delay slot with another instruction compared with the conventional delay branching technique can reduce the

useless clock consumption at the time of a branching instruction execution (branching penalty), and can shorten the processing time.

[0043] This invention is not limited to the operation gestalt mentioned above. For example, although single port memory with a singular read-out port was illustrated with the operation gestalt mentioned above as memory 80 shown in drawing 2 , a multiport memory with two or more read-out ports may be used. Moreover, with the operation gestalt mentioned above, as shown in drawing 1 , the configuration which performs five steps of pipe run processings was illustrated, but this invention can be applied also when performing five or more steps of pipeline processing. Furthermore, if the configuration of the instruction memory 61 shown in drawing 1 has the same function, it will not be limited to the configuration shown especially in drawing 2.

[0044]

[Effect of the Invention] As explained above, according to this invention, in pipeline processing, decline in the processing effectiveness by branch instruction can be controlled effectively.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]

[Drawing 1] Drawing 1 is the block diagram of an involving-in operation gestalt of this invention processor.

[Drawing 2] Drawing 2 is the block diagram of the instruction memory shown in drawing 1 .

[Drawing 3] Drawing 3 is drawing for explaining the storage gestalt of an instruction in the memory shown in drawing 2 .

[Drawing 4] Drawing 4 is drawing for explaining processing in case branching processing by branch instruction is performed in the pipeline processing of a processor shown in drawing 1 .

[Drawing 5] Drawing 5 is drawing for explaining processing in case branching processing by branch instruction is not performed in the pipeline processing of a processor shown in drawing 1 .

[Drawing 6] Drawing 6 is the block diagram of the processor of the conventional computer.

[Drawing 7] Drawing 7 is drawing for explaining the pipeline processing in the processor shown in drawing 6 .

[Drawing 8] Drawing 8 is drawing for explaining processing when branch instruction is executed in the pipeline processing shown in drawing 7 .

[Description of Notations]

41 -- A processor, 42 -- IF module, 43, 45, 47, 49 -- Register, 46 -- EX module, 48 -- An MEM module, 50 -- WB module, 60 -- A program counter, 61 -- Instruction

memory, 62 -- Multiplexer, 65 [-- Branching judging circuit,] -- A decoder, 66 -- A register file, 67 -- ALU, 68 69 -- Data memory, 80 -- Memory and 801-808 -- [-- An address register and 841-848 / -- The access-control section 851-858, 86, 87 -- Multiplexer] A bank, 81 -- 82 A flag register, 83

[Translation done.]

CN1206145 の対応JP

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－24929

(43)公開日　平成11年(1999)１月29日

(51)Int.Cl.[6]	識別記号	ＦＩ		
G06F 9/38	330	G06F 9/38	330	F

審査請求　未請求　請求項の数11　ＯＬ　（全９頁）

(21)出願番号　特願平9－174407

(22)出願日　平成９年(1997)６月30日

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号

(72)発明者　花木　博一
東京都品川区北品川６丁目７番35号　ソニー株式会社内

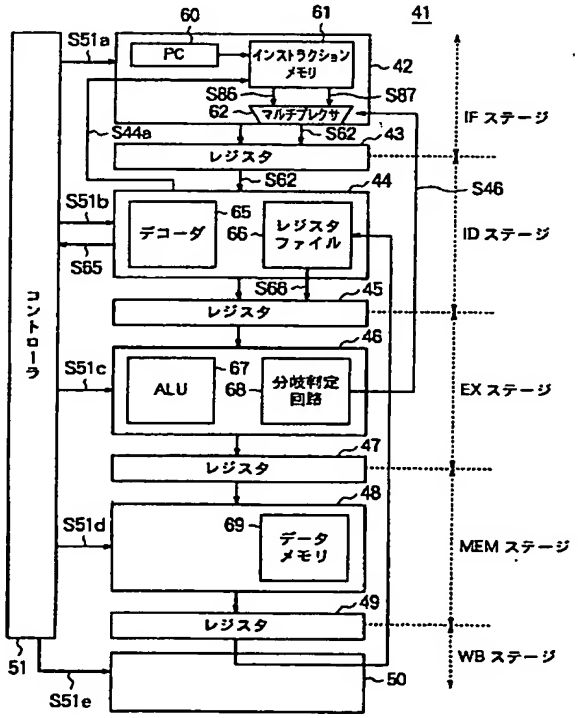
(74)代理人　弁理士　佐藤　隆久

(54)【発明の名称】演算処理装置およびその方法

(57)【要約】

【課題】　パイプライン処理において、分岐命令による処理効率の低下を効果的に抑制できる演算処理装置を提供する。

【解決手段】　ＩＤモジュール４４において、命令をデコードした結果、分岐命令であると認識した場合に、次のサイクルで、ＥＸモジュール４６において分岐の有無を判定すると共に、ＩＦモジュール４２において分岐先の命令と非分岐先の命令とを同時にフェッチする。そして、次のサイクルで、分岐の有無の判定結果に基づいて、フェッチした分岐先の命令および非分岐先の命令のいずれかを選択し、この選択した命令をＩＤモジュール４４においてデコードする。



1

【特許請求の範囲】

【請求項１】少なくとも、フェッチ手段による命令記憶部に記憶された命令のフェッチ処理と、デコード手段による前記フェッチした命令のデコード処理と、演算手段による前記デコード処理の結果に応じた演算処理と、記憶処理手段による前記演算処理の結果の記憶処理とをパイプライン方式で行う演算処理装置において、

前記フェッチ手段は、

前記命令記憶部上のアドレスを、順次にインクリメントして指し示すプログラムカウンタと、

デコードされた命令が、分岐命令である場合に、当該分岐命令に含まれる分岐先のアドレスを記憶するアドレス記憶部と、

同時にアクセス可能な複数のバンク領域に命令を記憶する命令記憶部と、

デコードされた命令が分岐命令である場合に、前記プログラムカウンタによって指し示される前記命令記憶部上のアドレスに記憶された命令と、前記アドレス記憶部に記憶されたアドレスによって指し示される前記命令記憶部上のアドレスに記憶された命令とを同時にフェッチするフェッチ部と、

前記分岐命令についての分岐条件の判定結果に基づいて、前記同時にフェッチした命令のうち一方の命令を選択して前記デコード手段に出力する選択部とを有する演算処理装置。

【請求項２】前記命令記憶部には、分岐命令による分岐先の命令と、非分岐先の命令とが、異なるバンクに記憶してある請求項１に記載の演算処理装置。

【請求項３】前記命令記憶部には、順次に連続して処理される前記バンク領域の数の命令が、異なるバンク領域に記憶してある請求項１に記載の演算処理装置。

【請求項４】前記命令記憶部は、単数の読み出しポートを持つシングルポートメモリである請求項１に記載の演算処理装置。

【請求項５】前記フェッチ手段は、

前記アドレス記憶部に記憶されているアドレスが有効であるかを示すフラグを記憶するフラグ記憶部をさらに有し、

前記フェッチ部は、前記フラグ記憶部に記憶されたフラグが有効であることを示す場合のみ、前記アドレス記憶部に記憶されたアドレスによって指し示される前記命令記憶部上のアドレスに記憶された命令をフェッチする請求項１に記載の演算処理装置。

【請求項６】前記フェッチ手段の前記フェッチ部は、

前記アドレスの第１の領域によって前記バンク領域を特定し、前記アドレスの第２の領域によってバンク領域内のアドレスを特定する請求項１に記載の演算処理装置。

【請求項７】前記デコード手段は、

前記選択部によって選択された命令を解読して、命令実行のための制御信号を生成するデコード部と、

2

演算手段において用いられるデータを記憶するデータ記憶部とを有する請求項１に記載の演算処理装置。

【請求項８】前記演算手段は、

算術論理部と、

分岐命令の分岐条件を判定する分岐判定部とを有する請求項１に記載の演算処理装置。

【請求項９】記憶処理手段は、

前記演算処理の結果を内蔵するデータ記憶部に記憶する処理と、前記演算処理の結果を前記デコード手段のデータ記憶部に記憶する処理とを行う請求項１に記載の演算処理装置。

【請求項１０】単数の前記フェッチ手段、デコード手段、演算手段および記憶処理手段を有する請求項１に記載の演算処理装置。

【請求項１１】少なくとも、命令記憶部に記憶された命令のフェッチ処理と、前記フェッチした命令のデコード処理と、前記デコード処理の結果に応じた演算処理と、前記演算処理の結果の記憶処理とをパイプライン方式で行う演算処理方法において、

前記命令記憶部上のアドレスを順次にインクリメントして、非分岐先の命令のアドレスを指し示し、

デコードされた命令が、分岐命令である場合に、当該分岐命令に含まれる分岐先のアドレスを記憶し、

同時にアクセス可能な複数のバンク領域を備えた命令記憶部に、分岐先の命令と非分岐先の命令とを異なるバンク領域に記憶し、

デコードされた命令が分岐命令である場合に、前記指し示された前記命令記憶部上のアドレスに記憶された非分岐先の命令と、前記アドレス記憶部に記憶されたアドレスによって指し示される前記命令記憶部上のアドレスに記憶された分岐先の命令とを同時にフェッチし、

前記分岐命令についての分岐条件の判定結果に基づいて、前記同時にフェッチした命令のうち一方の命令を選択してデコードする演算処理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、演算処理装置およびその方法に関する。

【０００２】

【従来の技術】ＤＳＰ(Digital Signal Processor)などに内蔵されるＲＩＳＣ(Reduced Instruction Set Computer)プロセッサは、一般的に、以下に示すようにしてプログラムに応じた演算処理を行う。すなわち、プロセッサでは、例えば、プログラムに含まれる命令のそれぞれについて、インストラクションメモリから命令をフェッチする（ＩＦ）ステージ、フェッチした命令をデコード（解読）する（ＩＤ）ステージ、デコードした命令を実行する（ＥＸ）ステージ、メモリにアクセスする（ＭＥＭ）ステージおよびアクセスしたメモリに結果を書き込む（ＷＢ）ステージを順に行うことで、その命令の演算

処理を行っている。この場合に、次の命令をフェッチするタイミングを、先の命令についてのＷＢステージが終了したタイミングにすると、先の命令のフェッチを開始するタイミングから、次の命令のＷＢステージが終了するタイミングまで、ＩＦステージ、ＩＤステージ、ＥＸステージ、ＭＥＭステージおよびＷＢステージのそれぞれに費やされる時間の総和を２倍した時間が必要となる。図６は、従来のコンピュータのプロセッサ１のブロック図である。図６に示すように、プロセッサ１は、ＩＦモジュール２、レジスタ３、ＩＤモジュール４、レジスタ５、ＥＸモジュール６、レジスタ７、ＭＥＭモジュール８、レジスタ９、ＷＢモジュール１０およびコントローラ１１を有する。

【０００３】ＩＦモジュール２、ＩＤモジュール４、ＥＸモジュール６、ＭＥＭモジュール８およびＷＢモジュール１０は、それぞれＩＦステージ、ＩＤステージ、ＥＸステージ、ＭＥＭステージおよびＷＢステージを実行する。

【０００４】ここで、従来から、プロセッサ１では、単位時間当たりの演算量を増やすために、複数の命令について、上述した各ステージの演算を並列的に行うパイプライン処理が採用されている。パイプライン処理では、図７に示すように、各ステージの処理を１サイクル内に終了するようにし、命令を１サイクル毎に順にプロセッサに入力し、異なる命令のＩＦステージ、ＩＤステージ、ＥＸステージ、ＭＥＭステージおよびＷＢステージを並列に実行する。

【０００５】具体的には、プロセッサ１では、図７に示すように、命令ｎ～ｎ＋４を、１サイクル間隔でプロセッサ１に入力し、サイクル２０において、命令ｎのＷＢステージと、命令ｎ＋１のＭＥＭステージと、命令ｎ＋２のＥＸステージと、命令ｎ＋３のＩＤステージと、命令ｎ＋４のＩＦステージとを並列に行う。このように、５段のパイプライン処理を採用すると、パイプライン処理を採用しない場合に比べて、１サイクル単位の演算量を５倍に高めることができる。

【０００６】なお、上述したプロセッサ１では、５段のパイプライン処理を採用した場合について例示したが、命令の処理をさらに細分化して、各ステージの処理を単純化することで、クロック周波数を高め、単位時間当たりの演算量をさらに増やすことも可能である。

【０００７】

【発明が解決しようとする課題】上述したように、プロセッサ１では、図７に示すように、命令ｎについてＥＸステージを開始したときに、命令ｎ＋１のＩＤステージおよび命令ｎ＋２のＩＦステージを開始する。ところで、命令ｎが分岐命令であった場合、命令ｎが分岐命令であることはＩＤステージで認識されるが、分岐するか否か、すなわち分岐条件が満たされているか否かは、ＥＸステージにおいて命令ｎを処理して初めて判定される。従って、命令ｎが分岐命令であると判明したときには、命令ｎの後続の命令ｎ＋１，ｎ＋２は、既にフェッチされている。

【０００８】このとき、そのままパイプラインに、命令ｎ＋１，ｎ＋２を、流し続けると、非分岐先の命令（分岐命令の直後に配置された命令）が実行されてしまい、正しい動作を行うことができない。そのため、例えば、図８に示すように、ＥＸステージにおいて分岐すると判定された時に、既にフェッチした後続の命令ｎ＋１，ｎ＋２を廃棄して、次のサイクルから、分岐先の命令ｍ，ｍ＋１，．．を順にフェッチする。しかしながら、このように既にフェッチした命令を破棄すると、処理効率が低下してしまうという問題がある。例えば、図８に示す場合には、分岐の発生によって、２サイクル遅延する。このような問題を解決するために、分岐命令に後続する命令のうち、分岐の有無の判定結果とは無関係に必ず実行される命令を、分岐命令の直後に配置し、分岐の有無によって実行の有無が決定される命令を遅延して行う遅延分岐という手法を採用する場合がある。ここで、分岐命令に後続する命令のうち、分岐の有無とは無関係に実行される命令群を遅延スロットと呼ぶ。

【０００９】このような遅延分岐を採用した場合に、遅延スロットの命令の数が、分岐の発生によってフェッチされた後に廃棄される可能性のある命令の数以上であれば、遅延スロットを分岐命令の直後に埋め込むことができるが、そうでない場合には、分岐命令の直後に、何も行わないｎｏｐ(no operation)命令を埋め込む必要があり、処理効率が低下してしまうという問題がある。例えば、図７に示す例では、遅延スロットの命令数が２以上ないと、ｎｏｐ命令を埋め込む必要があり、処理効率が低下してしまう。

【００１０】また、ＩＤステージにおいて、分岐命令を認識したときにパイプラインを停止し、分岐するか否かが決定してから分岐先もしくは非分岐先の命令をフェッチしてパイプラインを再始動させる方法などもある。

【００１１】しかしながら、いずれの方法でも、分岐命令を実行（分岐の判定）するまでの間は、次にフェッチすべき命令を特定できないので、フェッチすべき命令を特定するまでの間、パイプラインが停止し、処理効率が下がってしまう。このようにパイプライン処理を採用したプロセッサ１では、分岐命令よる、分岐ペナルティーが存在し、これをいかに削減するかが、処理効率を高める上で重要である。

【００１２】この分岐ペナルティーを少しでも削減するために、分岐するか否かを予め予測しておく方法もあるが、予測が外れた場合に大きなペナルティーが生じる。また、予測するための回路を設けることで、プロセッサが大規模化してしまうという問題もある。また、分岐判定をＩＤステージ中に行って即座に分岐させる方法もあるが、その際、判定の対象となるデータが分岐命令の前

の命令で演算中（ＥＸステージ）であるなどの場合にクリティカルパスとなり、高速実装が困難である。

【００１３】本発明は、上述した従来技術に鑑みてなされ、パイプライン処理において、分岐命令による処理効率の低下を効果的に抑制できる演算処理装置およびその方法を提供することを目的とする。

【００１４】

【課題を解決するための手段】上述した従来技術の問題点を解決し、上述した目的を達成するために、本発明の演算処理装置は、少なくとも、フェッチ手段による命令記憶部に記憶された命令のフェッチ処理と、デコード手段による前記フェッチした命令のデコード処理と、演算手段による前記デコード処理の結果に応じた演算処理と、記憶処理手段による前記演算処理の結果の記憶処理とをパイプライン方式で行う演算処理装置であって、前記フェッチ手段は、前記命令記憶部上のアドレスを、順次にインクリメントして指し示すプログラムカウンタと、デコードされた命令が、分岐命令である場合に、当該分岐命令に含まれる分岐先のアドレスを記憶するアドレス記憶部と、同時にアクセス可能な複数のバンク領域に命令を記憶する命令記憶部と、デコードされた命令が分岐命令である場合に、前記プログラムカウンタによって指し示される前記命令記憶部上のアドレスに記憶された命令と、前記アドレス記憶部に記憶されたアドレスによって指し示される前記命令記憶部上のアドレスに記憶された命令とを同時にフェッチするフェッチ部と、前記分岐命令についての分岐条件の判定結果に基づいて、前記同時にフェッチした命令のうち一方の命令を選択して前記デコード手段に出力する選択部とを有する。ここで、前記命令記憶部には、分岐命令による分岐先の命令と、非分岐先の命令とが、異なるバンクに記憶してある。

【００１５】また、本発明の演算処理方法は、少なくとも、命令記憶部に記憶された命令のフェッチ処理と、前記フェッチした命令のデコード処理と、前記デコード処理の結果に応じた演算処理と、前記演算処理の結果の記憶処理とをパイプライン方式で行う演算処理方法であって、前記命令記憶部上のアドレスを順次にインクリメントして、非分岐先の命令のアドレスを指し示し、デコードされた命令が、分岐命令である場合に、当該分岐命令に含まれる分岐先のアドレスを記憶し、同時にアクセス可能な複数のバンク領域を備えた命令記憶部に、分岐先の命令と非分岐先の命令とを異なるバンク領域に記憶し、デコードされた命令が分岐命令である場合に、前記指し示された前記命令記憶部上のアドレスに記憶された非分岐先の命令と、前記アドレス記憶部に記憶されたアドレスによって指し示される前記命令記憶部上のアドレスに記憶された分岐先の命令とを同時にフェッチし、前記分岐命令についての分岐条件の判定結果に基づいて、前記同時にフェッチした命令のうち一方の命令を選択してデコードする。

【００１６】

【発明の実施の形態】以下、本発明の実施形態に係わるプロセッサについて説明する。図１は、本実施形態のプロセッサ４１のブロック図である。図１に示すように、プロセッサ４１は、例えば、ＩＦモジュール４２、レジスタ４３、ＩＤモジュール４４、レジスタ４５、ＥＸモジュール４６、レジスタ４７、ＭＥＭモジュール４８、レジスタ４９、ＷＢモジュール５０およびコントローラ５１を有する。

【００１７】ＩＦモジュール４２、ＩＤモジュール４４、ＥＸモジュール４６、ＭＥＭモジュール４８およびＷＢモジュール５０は、それぞれＩＦステージ、ＩＤステージ、ＥＸステージ、ＭＥＭステージおよびＷＢステージを実行する。プロセッサ４１は、前述したプロセッサ１と同様に、パイプライン処理を行うが、分岐命令の処理がプロセッサ１とは異なる。すなわち、プロセッサ４１では、プロセッサ１と同様に、図６に示すように、各ステージの処理を１サイクル内に終了するようにし、命令を順に１サイクル毎にプロセッサに入力し、５個の命令のＩＦステージ、ＩＤステージ、ＥＸステージおよびＭＥＭステージを並列に実行してパイプライン処理を行う。また、プロセッサ４１では、プロセッサ１とは異なり、ＩＤモジュール４４において、命令をデコードした結果、分岐命令であると認識した場合に、次のサイクルで、ＥＸモジュール４６において分岐の有無を判定すると共に、ＩＦモジュール４２において分岐先の命令と非分岐先の命令とを同時にフェッチする。そして、次のサイクルで、分岐の有無の判定結果に基づいて、フェッチした分岐先の命令および非分岐先の命令のいずれかを選択し、この選択した命令をＩＤモジュール４４においてデコードする。

【００１８】以下、図１に示すプロセッサ４１の構成要素について詳細に説明する。先ず、ＩＦモジュール４２について説明する。図１に示すように、ＩＦモジュール４２は、例えば、プログラムカウンタ６０、インストラクションメモリ６１および選択部としてのマルチプレクサ６２を有する。プログラムカウンタ６０は、コントローラ５１からの制御信号Ｓ５１ａに基づいて、次に読み出しを行う命令のインストラクションメモリ６１上のアドレスを指し示し、１サイクル毎にアドレスを順次にインクリメントする。

【００１９】図２は、インストラクションメモリ６１のブロック図である。図２に示すように、インストラクションメモリ６１は、命令記憶部としてのメモリ８０、フラグレジスタ８１、アドレスレジスタ８２，８３、フェッチ部としてのアクセス制御部８４$_1$～８４$_8$、マルチプレクサ８５$_1$～８５$_8$およびマルチプレクサ８６，８７を有する。メモリ８０は、例えば、バンク８０$_1$～８０$_8$の８個のバンクを有するシングルポートメモリであ

り、これら８個のバンクに同時にアクセスできるようになっている。このように、メモリ８０として、シングルポートメモリを用いることで、装置構成を小規模かつ安価なものにすることができる。なお、メモリ８０のバンク数は、２のべき乗にしておくのが好ましい。

【００２０】バンク８０$_1$～８０$_8$には、図３に示すように、プログラム中に命令１，２，３，４，５，６，７，８を順に、それぞれバンク８０$_1$，８０$_2$，８０$_3$，８０$_4$，８０$_5$，８０$_6$，８０$_7$，８０$_8$に記憶し、次に、命令９，．．．を、バンク８０$_1$からバンク８０$_8$に向かって順に記憶する。このようにすることで、分岐命令があった場合に、分岐先の命令と非分岐先の命令とが同一のバンクに記憶される確率は１／８になる。すなわち、分岐先の命令と非分岐先の命令とが同一のバンクに記憶されると、これらの命令を同時にフェッチできなくなってしまう。このように、分岐先の命令と非分岐先の命令とが同一のバンクに記憶されてしまった場合には、分岐命令あるいは分岐先の命令の何れか一方を、プログラムのセマンティクス（意味）が変わらないように、他の命令と入れ替える。それができない場合には、ｎｏｐ（空）命令を挿入して、分岐先の命令と非分岐先の命令とが同一のバンクに記憶されないように、命令の位置をずらす。その結果、分岐命令について、分岐先の命令と非分岐先の命令とを、異なるバンクに記憶することができ、これらの命令を同時に読み出すことが可能になる。

【００２１】このように、メモリ８０を８バンク構成にした場合、例えば、アドレスレジスタ８２，８３に記憶されたアドレスの下位３ビットが、バンク番号を示し、それより上位のビットが各バンクにおけるアドレスを示すようにする。バンク８０$_1$～８０$_8$は、アドレスレジスタ８２あるいは８３に記憶されたアドレスの下位３ビットが、それぞれ「０００」、「００１」、「０１０」、「０１１」、「１００」、「１０１」、「１１１」のときにアクティブになる。

【００２２】アドレスレジスタ８２は、プログラムカウンタ６０が指し示す非分岐先の命令が記憶されたメモリ８０上のアドレスを記憶する。アドレスレジスタ８３は、ＩＤモジュール４４から入力した、分岐先の命令が記憶されたメモリ８０上のアドレスを記憶する。このように、ＩＦモジュール４２では、メモリ８０の２バンクを同時にアクセスするために、２個のアドレスレジスタを備えている。フラグレジスタ８１は、アドレスレジスタ８３に記憶された分岐先のアドレスが有効であるか否かを示すフラグを記憶する。フラグレジスタ８１には、ＩＤモジュール４４からアドレスレジスタ８３に分岐先のアドレスを記憶するときに「１」を示すフラグが記憶され、それ以外のときには、「０」を示すフラグが記憶される。

【００２３】マルチプレクサ８５$_1$～８５$_8$は、例えば、コントローラ５１からの制御信号Ｓ５１ａに基づいて、アドレスレジスタ８２に記憶された非分岐先の命令のアドレスと、アドレスレジスタ８３に記憶された分岐先の命令のアドレスとのうち何れか一方を選択して、それぞれアクセス制御部８４$_1$～８４$_8$に出力する。アクセス制御部８４$_1$～８４$_8$は、それぞれ、マルチプレクサ８５$_1$～８５$_8$からのアドレスに基づいて、そのアドレスの下位３ビットが対応するバンク８０$_1$～８０$_8$を示している場合には、そのアドレスの残りの上位ビットを用いて、バンク８０$_1$～８０$_8$から命令を読み出す。また、アクセス制御部８４$_1$～８４$_8$は、フラグレジスタ８１に記憶されたフラグが「１」の場合には、アドレスレジスタ８３に記憶されたアドレスによるバンク８０$_1$～８０$_8$への読み出し動作は行わない。

【００２４】マルチプレクサ８６は、アクセス制御部８４$_1$～８４$_8$からの読み出し結果のうち、アドレスレジスタ８２に記憶されたアドレスの下位３ビットで指定されるバンク８０$_1$～８０$_8$からの読み出し結果を選択し、この選択された分岐先命令Ｓ８６をマルチプレクサ６２に出力する。マルチプレクサ８６は、アクセス制御部８４$_1$～８４$_8$からの読み出し結果のうち、アドレスレジスタ８３に記憶されたアドレスの下位３ビットで指定されるバンク８０$_1$～８０$_8$からの読み出し結果を選択し、この選択された非分岐先命令Ｓ８７をマルチプレクサ６２に出力する。

【００２５】ＩＦモジュール４２では、アドレスレジスタ８２に記憶されたアドレスで指し示される非分岐先命令のアドレスと、アドレスレジスタ８２に記憶されたアドレスで指し示される分岐先命令とを同時に読み出しておく。このとき、分岐命令はＥＸステージ上にあり分岐するか否かを判定中である。この判定サイクルが終わる前にはＥＸモジュール４６から分岐判定結果Ｓ４６がマルチプレクサ６２に返されるので、その結果により、同時に読み出しておいた分岐先命令Ｓ８６あるいは非分岐先命令Ｓ８７のいずれかをマルチプレクサ６２において選択し、ＩＦモジュール４２の処理を終了する。この選択された命令Ｓ６２は、図１に示すレジスタ４３でラッチされた後に、ＩＤモジュール４４に出力される。

【００２６】次に、図１に示すＩＤモジュール４４について説明する。ＩＤモジュール４４は、図１に示すように、デコーダ６５およびレジスタファイル６６を有する。デコーダ６５は、制御信号Ｓ５１ｂに基づいて、レジスタ４３を介してＩＦモジュール４２から入力した命令Ｓ６２をデコードし、命令実行のための各種の制御信号を作成し、この制御信号Ｓ６５をコントローラ５１に出力すると共に、レジスタファイル６６にアクセスし、ＥＸモジュール４６における演算に用いるデータを読み出し、この読み出したデータＳ６６をレジスタ４５でラッチした後に、後段のＥＸモジュール４６に出力する。また、デコーダ６５は、レジスタ４３からの命令Ｓ６２

9

をデコードした結果、分岐命令であった場合に、分岐先のアドレスＳ４４ａを図２に示すＩＦモジュール４２のアドレスレジスタ８３に出力して記憶させると共に、ＩＦモジュール４２のフラグレジスタ８１に「１」を示すフラグを記憶させる。これにより、次のサイクルでは、この分岐命令がＥＸモジュール４６において分岐判定されると同時に、ＩＦモジュール４２において分岐先命令と非分岐先命令とが同時に読み出される。

【００２７】次に、図１に示すＥＸモジュール４６について説明する。ＥＸモジュール４６は、演算処理を行うＡＬＵ(Arithmetic and Logic Unit)６７、分岐判定回路６８および図示しないアドレス生成回路を有する。ＡＬＵ６７は、コントローラ５１からのデコード結果に応じた制御信号Ｓ５１ｃに基づいて、データＳ６６を用いて、演算処理を行う。アドレス生成回路は、ＡＬＵ６７の演算処理結果のデータを記憶するデータメモリ６９上のアドレスを生成する。なお、ＡＬＵ６７は、データメモリ６９に記憶されたデータを、レジスタファイル６６に読み出した後に、レジスタファイル６６にアクセスを行なって、そのデータを演算処理に用いる。また、ＡＬＵ６７は、レジスタファイル６６を介して、演算処理結果をデータメモリ６９に記憶する。

【００２８】ＥＸモジュール４６は、ＡＬＵ６７の演算処理結果と、アドレス生成回路が生成したアドレスとをレジスタ４７を介してＭＥＭモジュール４８に出力する。分岐判定回路６８は、ＡＬＵ６７において演算中の命令が分岐命令であり、かつ、分岐条件を評価した結果、分岐すると判定した場合に、分岐することを指示する分岐判定結果Ｓ４６をＩＦモジュール４２に出力する。ＩＦモジュール４２は、分岐判定結果Ｓ４６に基づいて、同時にフェッチした分岐先命令もしくは非分岐命令のいずれかを図２に示すマルチプレクサ６２で選択して出力する。

【００２９】次に、ＭＥＭモジュール４８について説明する。ＭＥＭモジュール４８は、データメモリ６９および図示しない制御回路を有する。ＭＥＭモジュール４８は、書込命令の場合には、コントローラ５１からの制御信号Ｓ５１ｄに基づいて、レジスタ４７を介してＥＸモジュール４６から入力したデータメモリ６９上のアドレスに、ＥＸモジュール４６から入力した演算処理結果のデータを記憶する（書き込む）。

【００３０】ＭＥＭモジュール４８は、読出命令の場合には、コントローラ５１からの制御信号Ｓ５１ｄに基づいて、レジスタ４７を介してＥＸモジュール４６から入力したデータメモリ６９上のアドレスから、データを読み出す。また、ＭＥＭモジュール４８は、データメモリ６９にアクセスを行わない命令の場合には、レジスタ４７を介してＥＸモジュール４６から入力いた演算処理結果のデータを、レジスタ４９を介して、そのままＷＢモジュール５０に出力する。

10

【００３１】さらに、ＭＥＭモジュール４８は、データメモリ６９から読み出したデータ、あるいは、ＥＸモジュール４６からの演算結果のデータのいずれかを、コントローラ５１からの制御信号に基づいて、マルチプレクサで選択し、レジスタ４９を介して、ＷＢモジュール５０に出力する。

【００３２】次に、ＷＢモジュール５０について説明する。ＷＢモジュール５０は、制御信号Ｓ５１ｅに基づいて、レジスタ４９を介して、ＭＥＭモジュール４８から入力したデータを、ＩＤモジュール４４のレジスタファイル６６に記憶する。

【００３３】以下、プロセッサ４１の動作について説明する。図４は、プロセッサ４１において分岐命令による分岐が発生した場合のパイプライン処理を説明するための図である。先ず、サイクル「１」において、図１に示すＩＦモジュール４２で命令ｎがフェッチされ、次のサイクル「２」において、ＩＤモジュール４４で命令ｎのデコードが行われると共に、ＩＦモジュール４２で命令ｎ＋１のフェッチが行われる。このとき、ＩＦモジュール４２では、図２に示すフラグレジスタ８１に「０」を示すフラグが記憶されており、図２に示すアクセス制御部８４１～８４８は、アドレスレジスタ８２に記憶されたプログラムカウンタ６０によって指し示されるアドレスに基づいて、メモリ８０から命令を読み出し、この読み出した命令をマルチプレクサ８６，６２を介して、レジスタ４３に出力する。また、ＩＤモジュール４４において、命令ｎが分岐命令であることが認識され、図１に示すデコーダ６５は、図２に示すインストラクションメモリ６１のフラグレジスタ８１に「１」を示すフラグを記憶すると共に、アドレスレジスタ８３に分岐先の命令のアドレスを記憶する。

【００３４】次に、図４に示すサイクル「３」において、ＥＸモジュール４６の分岐判定回路６８で、命令ｎの分岐条件が満たされているか否かが判定され、例えば、分岐条件が満たされている場合に、そのことを示す分岐判定結果Ｓ４６を図１および図２に示すマルチプレクサ６２に出力する。それと同時に、図２に示すインストラクションメモリ６１において、アドレスレジスタ８２およびアドレスレジスタ８３に記憶されたアドレスに基づいて、アクセス制御部８４１～８４８において、メモリ８０から分岐先命令ｍおよび非分岐先命令ｎ＋２が読み出される。そして、非分岐先命令ｎ＋１（Ｓ８６）および分岐先命令ｍ（Ｓ８７）が、マルチプレクサ６２に出力され、マルチプレクサ６２において、分岐判定結果Ｓ４６に基づいて、分岐先命令ｍが選択され、命令Ｓ６２としてレジスタ４７を介して、ＭＥＭモジュール４８に出力される。また、サイクル「２」においてＩＦモジュール４２でフェッチされた命令ｎ＋１は破棄される。

【００３５】次に、サイクル「４」において、ＭＥＭモ

ジュール４８、ＩＤモジュール４４およびＩＦモジュール４２で、それぞれ命令ｎのＭＥＭステージ、命令ｍのＩＤステージ、命令ｍ＋１のＩＦステージが行われる。

【００３６】次に、サイクル「５」において、ＷＢモジュール５０、ＥＸモジュール４６、ＩＤモジュール４４およびＩＦモジュール４２において、命令ｎのＷＢステージ、命令ｍのＥＸステージ、命令ｍ＋１のＩＤステージおよび命令ｍ＋２のＩＦステージがそれぞれ行われる。以下、同様に、分岐命令が存在しない限り、命令ｍ＋３，ｍ＋４，．．について、ＩＦステージ、ＩＤステージ、ＥＸステージ、ＭＥＭステージおよびＷＢステージが順次に行われる。

【００３７】図５は、プロセッサ４１において分岐命令による分岐が発生しない場合のパイプライン処理を説明するための図である。この場合には、サイクル「１」，「２」は、前述した図４に示す分岐命令による分岐が発生する場合のパイプライン処理と同様の処理が行われる。

【００３８】次に、サイクル「３」において、ＥＸモジュール４６の分岐判定回路６８で、命令ｎの分岐条件が満たされているか否かが判定され、例えば、分岐条件が満たされていない場合に、そのことを示す分岐判定結果Ｓ４６を図１および図２に示すマルチプレクサ６２に出力する。それと同時に、図２に示すインストラクションメモリ６１において、アドレスレジスタ８２およびアドレスレジスタ８３に記憶されたアドレスに基づいて、アクセス制御部８４１～８４ｓにおいて、メモリ８０から分岐先命令ｍおよび非分岐先命令ｎ＋２が読み出される。そして、非分岐先命令ｎ＋２（Ｓ８６）および分岐先命令ｍ（Ｓ８７）が、マルチプレクサ６２に出力され、マルチプレクサ６２において、分岐判定結果Ｓ４６に基づいて、非分岐先命令ｎ＋２が選択され、命令Ｓ６２としてレジスタ４７を介して、ＭＥＭモジュール４８に出力される。また、サイクル２において、ＩＦモジュール４２においてフェッチされた命令ｎ＋１は破棄される。

【００３９】次に、サイクル「４」において、ＭＥＭモジュール４８、ＩＤモジュール４４およびＩＦモジュール４２で、それぞれ命令ｎのＭＥＭステージ、命令ｎ＋１のＩＤステージ、命令ｎ＋２のＩＦステージが行われる。

【００４０】次に、サイクル「５」において、ＷＢモジュール５０、ＥＸモジュール４６、ＩＤモジュール４４およびＩＦモジュール４２において、命令ｎのＷＢステージ、命令ｎ＋１のＥＸステージ、命令ｎ＋２のＩＤステージおよび命令ｎ＋３のＩＦステージがそれぞれ行われる。以下、同様に、分岐命令が存在しない限り、命令ｎ＋４，ｎ＋５，．．について、ＩＦステージ、ＩＤステージ、ＥＸステージ、ＭＥＭステージおよびＷＢステージが順次に行われる。

【００４１】以上説明したように、プロセッサ４１によれば、ＩＤモジュール４４において命令が分岐命令であると認識された場合、次のサイクルで、ＥＸモジュール４６において、この分岐命令が実行され分岐判定をしている間に、ＩＦモジュール４２では分岐先命令と非分岐先命令の両方を同時に読み出してあり、分岐判定結果Ｓ４６が得られ次第、該当する命令を選択する。そのため、分岐判定結果Ｓ４６が分岐および非分岐の何れであろうと、その次のサイクルには分岐先命令あるいは非分岐先命令をＩＤモジュール４４に出力することができる。そのため、前述した従来の並列プロセッサ１に比べて、分岐発生による処理効率の低下を効果的に抑制できる。

【００４２】具体的には、プロセッサ４１によれば、従来の分岐予測を行わない手法に比べると、分岐命令出現回数に応じたサイクルだけ、処理時間を短縮することができる。また、プロセッサ４１によれば、従来の分岐予測を行う手法に比べると、分岐予測が外れた回数に応じたサイクルだけ、処理時間を短縮することができる。また、プロセッサ４１によれば、従来の遅延分岐手法と比べると、遅延スロットを別の命令で埋められないために挿入されたｎｏｐ命令の数だけ分岐命令実行時の無駄なクロック消費（分岐ペナルティー）を削減できることになり、処理時間を短縮できる。

【００４３】本発明は上述した実施形態には限定されない。例えば、上述した実施形態では、図２に示すメモリ８０として、単数の読み出しポートを持つシングルポートメモリを例示したが、複数の読み出しポートを持つマルチポートメモリを用いてもよい。また、上述した実施形態では、図１に示すように、５段のパイプラン処理を行う構成を例示したが、５段以上のパイプライン処理を行う場合にも、本発明を適用できる。さらに、図１に示すインストラクションメモリ６１の構成は、同様の機能を持つものであれば、特に図２に示す構成には限定されない。

【００４４】

【発明の効果】以上説明したように、本発明によれば、パイプライン処理において、分岐命令による処理効率の低下を効果的に抑制できる。

【図面の簡単な説明】

【図１】図１は、本発明の実施形態に係わるのプロセッサのブロック図である。

【図２】図２は、図１に示すインストラクションメモリのブロック図である。

【図３】図３は、図２に示すメモリへの命令の記憶形態を説明するための図である。

【図４】図４は、図１に示すプロセッサのパイプライン処理において、分岐命令による分岐処理が実行される場合の処理を説明するための図である。

【図５】図５は、図１に示すプロセッサのパイプライン

13

処理において、分岐命令による分岐処理が実行されない場合の処理を説明するための図である。

【図６】図６は、従来のコンピュータのプロセッサのブロック図である。

【図７】図７は、図６に示すプロセッサにおけるパイプライン処理を説明するための図である。

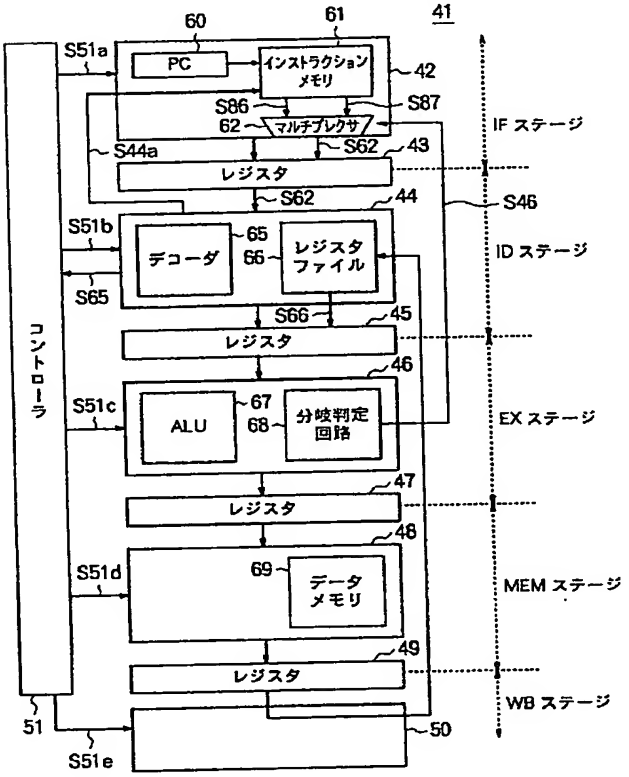
【図８】図８は、図７に示すパイプライン処理において分岐命令が実行された場合の処理を説明するための図である。

【符号の説明】

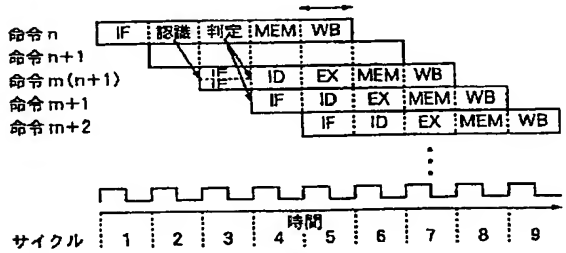
14

４１…プロセッサ、４２…ＩＦモジュール、４３，４５，４７，４９…レジスタ、４６…ＥＸモジュール、４８…ＭＥＭモジュール、５０…ＷＢモジュール、６０…プログラムカウンタ、６１…インストラクションメモリ、６２…マルチプレクサ、６５…デコーダ、６６…レジスタファイル、６７…ＡＬＵ、６８…分岐判定回路、６９…データメモリ、８０…メモリ、80_1～80_8…バンク、８１…フラグレジスタ、８２，８３…アドレスレジスタ、84_1～84_8…アクセス制御部、85_1～85_8，８６，８７…マルチプレクサ

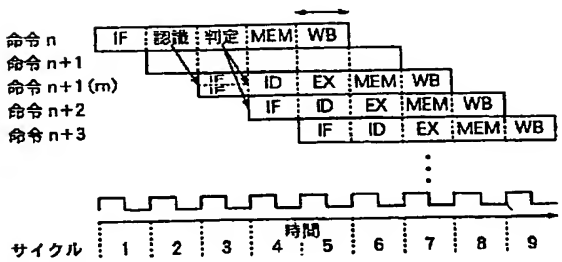
【図１】



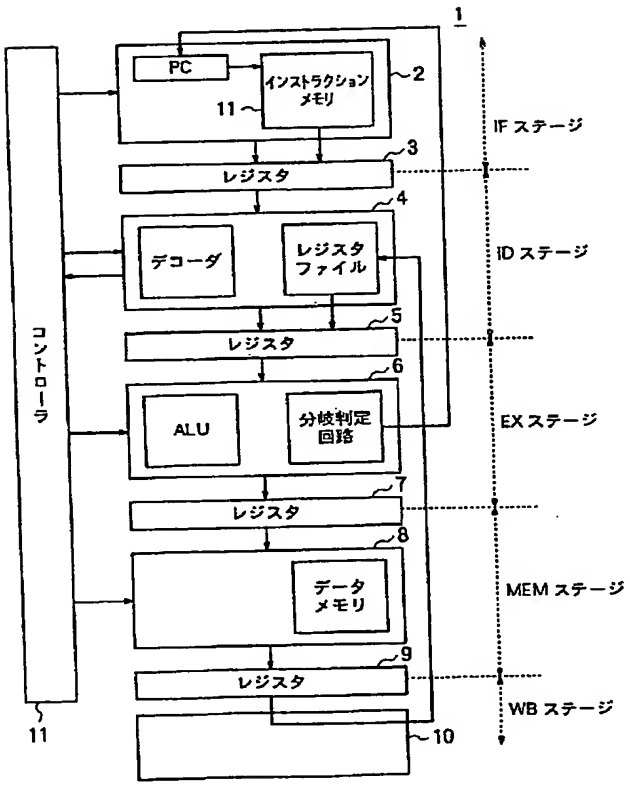
【図２】



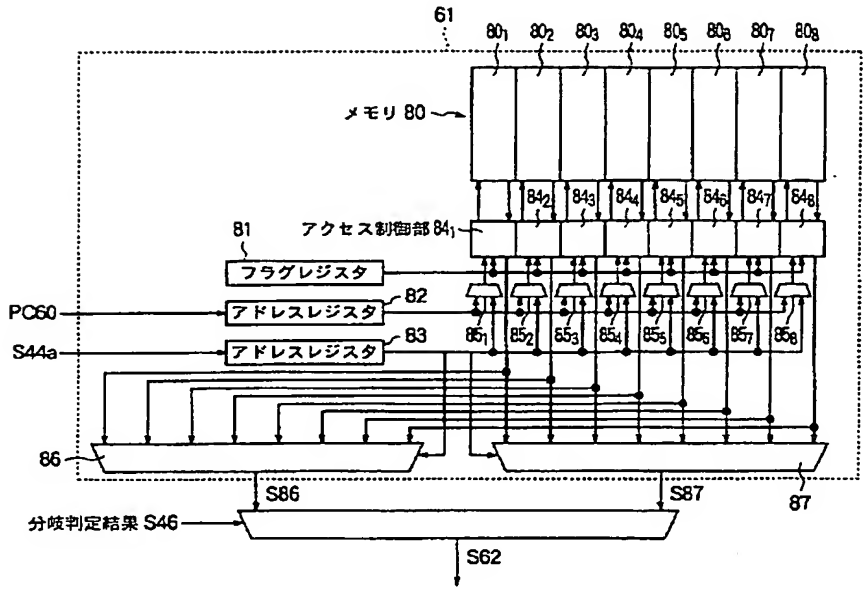
【図５】



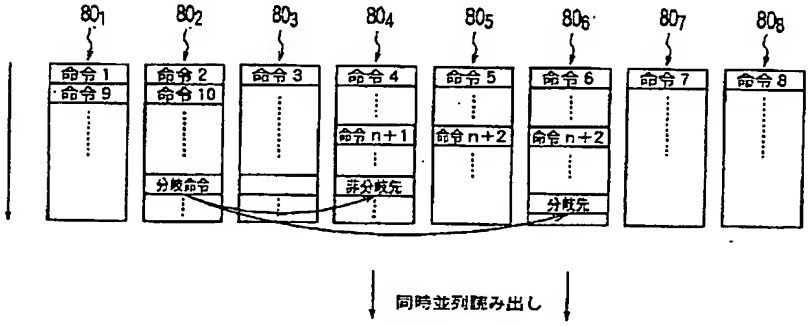
【図６】



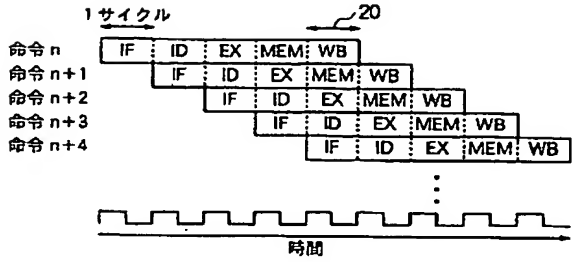
【図２】


61
メモリ 80
81
アクセス制御部 84
フラグレジスタ
PC60
アドレスレジスタ
82
S44a
アドレスレジスタ
83
86
S86
S87
87
分岐判定結果 S46
S62

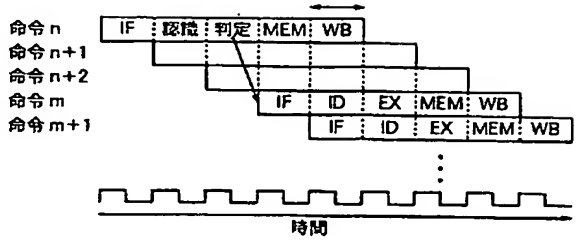
【図３】


命令 1
命令 9
命令 2
命令 10
命令 3
命令 4
命令 n+1
命令 5
命令 n+2
命令 6
命令 n+2
命令 7
命令 8
分岐命令
非分岐先
分岐先
同時並列読み出し

【図７】


1サイクル
20
命令 n
命令 n+1
命令 n+2
命令 n+3
命令 n+4
IF
ID
EX
MEM
WB
時間

【図８】


命令 n
命令 n+1
命令 n+2
命令 m
命令 m+1
IF
認識
判定
MEM
WB
ID
EX
時間